TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Reimo Jaansalu 185750IADB

Karakteri riietusesemete efektide planeerija veebimängule

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Reimo Jaansalu

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus arvutimängule nimega Growtopia.

Arendusprotsessi käigus luuakse veebirakendus, mille põhiülesanne on teha lihtsamaks mängijatele erinevate mängus olevate riideesemete ja nende efektide otsimine ja filtreerimine ning pakkuda simuleeritud karakteri riietamise võimalus, millega saab näha, mis efekte saab valitud riideesemete kandmisel.

Töö rõhk on kasutajamugavusel ning teenuse ja kliendi eraldatusel, et soodustada uute funktsionaalsuste lisamist ning teiste mängu olemasolevate rakendustega liidestamist. Arendusprotsessi tulemuseks on töötav veebirakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 6 peatükki, 4 joonist, 1 tabelit.

Abstract

Character Clothing Effects Planner for a Video Game

The aim of the current thesis is to create a web application for the video game called Growtopia.

During the development process, a web application will be created, the main task of which is to make it easier for players to search and filter different in-game clothing items and their effects, and to offer the possibility of equipping a simulated character to see what effects can be obtained by wearing the selected items.

The emphasis of the work is on user experience and the separation of the service and the client in order to encourage the addition of new functionalities and interfacing with other existing applications that are done for the game. The development process results in a working web application.

The thesis is in and contains 38 pages of text, 6 chapters, 4 figures, 1 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
backend	Teenusepoolne keskkond
CSS	*Cascading Style Sheets* - veebilehtede kujundamisel kasutatav märgistuskeel
frontend	Kasutajaliides, kliendipoolne keskkond
HTML	*Hyper Text Markup Language* - veebilehe märgendamise keel
JSON	*Javascript Object Notation* - Javascript'il põhinev andmevahetusvorming
ORM	*Object-Relational Mapping* – põhimõte, mille eesmärk on objekt-orienteeritud koodi kaardistada ümber teise andmevormi
RDBMS	*Relational Database Management System* – relatsiooniline andmebaasi haldussüsteem
REST	*Representational State Transfer* – veebiteenusega suhtlemise liidese arhitektuur
URL	*Uniform Resource Locator*, veebiaadress
XML	*Extensible Markup Language* – dokumentide märgendamiskeel, mis on nii masin- kui ka inimloetav

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Growtopia on populaarne 2D veebipõhine “liivakasti” mitmikmäng, s.t mängul pole kindlat lõppeesmärki, ning seda saab mängida nii arvutis kui ka mobiilsete seadmete peal koos teiste inimestega (ei pea olema sama seadme taga). Mängijad saavad luua enda või külastada teiste virtuaalmaailmaid ja saavad nendes ehitada, kasvatada puid või loomi, suhelda ja vahetada esemeid teiste mängijatega, osaleda igasugustes tegevustes, mis on kas mängu sisseehitatud või teiste mängijate poolt loodud. Olgu nendeks põlluharimine, kaevandamine, operatsioonide tegemine, parkuurimine, võistlustes osalemine, esemete müümine ja palju muud.

Mängu mängimiseks juhitakse karakterit, kellele saab igasuguseid riideesemeid selja panna ja mõned nendest annavad erilisi efekte. Olgu nendeks efektideks lisa kiirus, kõrgemale hüppamine, kiiremini kaevamine jne. Need efektid on mängus igapäevasteks tegevusteks väga suureks abiks, kui mitte hädavajalikud, või annavad suuri eeliseid teiste mängijate suhtes.

Mängus on väga palju riideesemeid ja efekte, ning on raske leida mingiks tegevuseks või olukorraks vajalikke efekte. Tihti ka ei teata, mis efektid olemas on, kuna mäng soodustab avastamist. Mängule on tehtud oma vikipeedia, kus on küll kõik esemed koos efektidega olemas, kuid seal on raske avastada enda jaoks midagi uut.

Käesolevas lõputöös luuakse veebirakendus, kus saab paremini näha, otsida ja filtreerida riideesemeid, efekte ja näha nende infot. Samuti saab rakenduses olema simuleeritud karakter, keda saab nii-öelda riietada, mille käigus saab näha, mis efekte karakter neid riideesemeid kandes saab.

2 Metoodika

Käesoleva lõputöö käigus räägitakse algul mängu eksisteerivast probleemist, uuritakse mängule tehtud muid olemasolevaid rakendusi ning töötatakse nõuete kohaselt välja uus rakendus, mis jääb antud lõputöö skoopi ning mis soodustaks selle edasi arendamise võimalusi tulevikus.

Lahenduse analüüsi käigus tuuakse välja nii funktsionaalsed kui ka mittefunktsionaalsed nõuded, analüüsitakse erinevaid tehnilisi lahendusi ning põhjendatakse teenuse- ja kliendipoolse tehnoloogiate ning ka arendus- ja haldusvahendite valikuid.

Lahenduse arenduse peatükid käsitlevad rakenduse kliendi- ja teenusepoolse rakenduse valmimist, kus räägitakse erinevatest lahenduse võtetest, andmebaasi disainist ja koodi struktuurist. Samuti käiakse üle rakenduse testimine.

Lõputöö viimastes peatükkides räägitakse rakenduse kasutatavusest ja edasi arendamise võimalustest väljaspool lõputöö skoopi.

3 Ülevaade probleemist

Growtopia mängu (https://www.growtopiagame.com) mõte on avastada, ehitada, suhelda, teha vahetuskaupa teiste mängijatega, osaleda endale meelepärastes tegevustes ja palju muud. Mängu igapäevasteks tegevusteks on väga suureks abiks erinevad efektid, näiteks kiiremini liikumine, kõrgemale hüppamine jne. Hetkel saab mängus efekte peamiselt teatud plokkidest, ühekordselt kasutatavatest esemetest ning riideesemetest. Kõige rohkem saab neid riideesemetest. Probleemiks on see, et neid on tohutult palju ning kui mängija ei ole kursis iga mängu eseme ja uuendusega, ei tea ta, mis esemed ja efektid olemas on. Ka vanematel mängijatel, kes on pidevalt kursis mänguga, võivad ununeda esemed koos efektidega. Lisaks on mängus efektid, mis on eri nimetusega, kuid käituvad identselt - see võib tekitada segadust. Nende tulemusena võib mängija teha tegevusi, kus jäävad efektidega boonused saamata.

Kuna mäng soodustab avastamist, siis mängu uuendamistel ei öelda mängijatele, mis esemed mängu lisati või olemas on. Seega peab mängija info saamiseks ise uued esemed avastama või lootma vikipeedia või teiste mängijate peale.

Vikipeedias, kus mängu infot uuendatakse mängijate poolt, on kõik mängu esemed olemas, kuid spetsiifilistele esemetele peale sattuda on raske ja teiste mängijate käest küsimine võib tihtipeale osutuda tülikaks. Seega pole hetkel ühtegi head esemete ja nende efektide otsimise võimalust.

3.1 Eksisteerivad lahendused

Käesoleva lõputöö koostamise hetkel ei ole antud probleemile otsest olemasolevat lahendust. Mängijad saavad otsida riideesemeid ja nende efekte mängu vikipeediast, kus on kõik info olemas, juhul, kui nad on juba otsitavast esemest teadlikud. Vikipeedias on ka tavaliselt mängu uuendustele pühendatud lehed infoga, kuid kui mängija ei ole iga uuendusega kursis, jäävad need tal avastamata.

Peale vikipeedia on mängule tehtud veel mõned rakendused. Populaarsemad neist on G*rowstocks* (https://growstocks.xyz) ja *Set-planner* (https://set-planner.growtopia.tech). *Growstocks* fokuseerib esemete mängusisestele hindadele, näidates valitud eseme hinda ja selle kõikumist ning ajalugu. Esemete hinnad põhinevad nende tarnele mängus ning

hinnad uuendatakse veebilehe haldajate poolt. *Set-planner* laseb rakenduse kasutajal riietada simuleeritud karakteri, otsides riideesemeid ning nende peale vajutades saab näha, kuidas valitud riided karakteri peal välja näevad. Peale visuaalse poole, riiete otsimise ja filtreerimise kehaosa järgi pole rakenduses ühtegi muud funktsionaalsust.

3.2 Uue lahenduse skoop

Kuna mängule on olemas rakendused, mis näitavad esemete hinda ja visuaalset poolt karakteril, jäävad need aspektid loodavast rakendusest välja. Antud lõputöö autor keskendub ainult riideesemetele, efektidele ja nende otsimisele, simuleeritud karakterile ning vajaliku info kuvamisele. Rakendus saab olema standardne REST veebirakendus. Kuna mäng on mängitav nii arvutis kui ka mobiilsetes seadmetes peab rakendus olema kasutatav kõikides seadmetes. Seega saab kohaneva rakenduse lähenemist kasutada ehk veebirakendus kohaneb vastavalt ekraani suurusele ja muudab vajadusel elementide paiknemist. Andmed saadakse andmebaasi mängu vikipeediat veebi-kraapides ning andmed lisatakse SQL lausetega. Samuti saab rakendus olema tulevikukindel, lubades uusi funktsionaalsusi lisada ilma suurema vaevata.

3.3 Uue lahenduse ärilise tasuvuse analüüs

Kuna tegemist on siiski hobiga seotud rakendusega ja lahendus on suhteliselt lihtne ühe lehe rakendus, ei ole sellel palju ärilise tasuvuse variante. Sellegipoolest variandid, mida saaks kasutada, oleks järgmised:

- Reklaamid - rakenduse vaba pinna asendamine reklaamidega näiteks Google AdSense [1]
- Mängusisesed reklaamid - osad mängijad võivad tahta oma maailma, teenuseid vms esile tuua, andes võimaluse neil seda tasuliselt teha
- Annetused - inimeste heast tahtest annetatud raha. Selleks kasutada lihtsat rahakogumise teenust näiteks PayPal või Patreon [1]

Parim valik oleks kasutada kõiki olemasolevaid variante, mis võiks katta minimaalselt võimalikud kulud.

4 Loodava veebirakenduse analüüs

Järgnevates peatükkides on kirjeldatud loodava rakenduse nõuded, kaalutud tehnoloogia, andmebaasi ja arenduskeskkonna valik ning räägitud lähemalt rakenduse ja andmebaasi disainist.

4.1 Nõuete määramine

Nõuete määramisel on suurimaks osakaaluks lõputöö autori enda kogemus mängust. Otsimine, filtreerimine ja info kuvamine peaks olema lihtne, selge ja kaasaegne.

4.1.1 Funktsionaalsed nõuded

Kasutaja nõuded:

- Kasutajana tahan näha kõiki riideesemete jaoks olevad kehaosi
- Kasutajana tahan näha kõiki efekte ja võimalus neid otsida
- Kasutajana tahan näha kõiki efekte valitud kehaosale ja võimalus neid otsida
- Kasutajana tahan näha efektide kirjeldust, nende kuhjumise maksimaalset väärtust
- Kasutajana tahan näha kõiki efektidega riideesemeid ja võimalus neid otsida
- Kasutajana tahan näha kõiki efektidega riideesemeid valitud kehaosadele ja/või efektile ja võimalus neid otsida
- Kasutajana tahan näha, kui mingi riideese vajab mingisugust teist riideeset, et efekte saada
- Kasutajana tahan eristada mitte vahetatavaid riideesemeid
- Kasutajana tahan eristada riideesemeid, mille välimust saab muuta
- Kasutajana tahan simuleerida erinevate riideesemete kandmist, et näha, mis efekte neid kandes saab

4.1.2 Mittefunktsionaalsed nõuded

Kasutaja põhised mittefunktsionaalsed nõuded:

- Kasutajana tahan, et veebirakendus oleks samas keeles, mis originaalmäng, s.t inglise keeles
- Kasutajana tahan, et veebirakendus oleks visuaalselt sarnane originaal mänguga

- Kasutajana tahan, et veebirakendus oleks võimalik kasutada ka mobiili peal
- Kasutajana tahan, et rakenduse võrgukasutus ei oleks liialt suur

4.2 Tehnoloogia valik

Veebirakenduse arendamiseks on valikus mitmeid erinevaid tehnoloogiaid. Antud lõputöö raames on kaalutud mõned enamlevinud valikud nimelt REST, SOAP ja GraphQL [2].

REST - lühend terminile REpresentational State Transfer. See laseb kasutajatel teha API päringuid kasutades lihtsaid HTTP käske. Samuti ei hoia server olekut päringute vahel. REST toetab mitmeid erinevaid andmeformaate (näiteks JSON) ning on väga paindlik ja skaleeritav. Siiski võib suuremate rakenduste puhul jääda veidi aeglaseks - kui andmeid on palju ja kui kõike infot pole korraga vaja saada [3].

SOAP - lühend terminile Simple Object Access Protocol. See on sarnane REST tehnoloogiale, kuid kasutab tihtipeale XML formaati. Samuti on see hea privaatsuseks, krüpteerides serveri vahelist suhtlust. Kuid SOAP ei ole eriti kohandatav ning XMLi pärast on seda tehnoloogiat mõistlik kasutada ainult siis, kui on saadaval suuremad interneti ribalaiused [3].

GraphQL - on Facebooki loodud tehnoloogia. See on JSONi sarnase struktuuriga ja on lihtsasti kasutatav ning skaleeritav [3]. GraphQLi eripära on võimalus lasta klientidel täpsustada, mida nad oma päringuga tahavad saada. Samuti saab ühe päringuga saada infot mitmest allikast. GraphQL on ka hea privaatsuse jaoks, jättes olulise info krüpteerituna [4]. Siiski pole see parim valik, kui on vaja teha suuri ja keerulisi päringuid.

Kõik eelpool nimetatud valikud on head antud lõputöö rakenduse teostamiseks, Kuna loodav rakendus ei ole mahukas ega vaja andmete krüpteerimist, sobib antud lahenduseks kõige paremini REST tehnoloogia. Suurenisti mõjutas valikut ka lõputöö autori kogemus sellega.

4.2.1 Teenusepoolse programmeerimiskeele valik

Tänapäeval on teenusepoolse programmeerimiskeele valik üpriski rikkalik. Nimelt on kaalutud antud lõputöös järgmisi enamlevinud [5], [6] keeli:

Javascript - kõige laialdaselt levinud programmeerimiskeel, mida kasutatakse nii teenuse- kui ka kliendipoolse osa jaoks. Javascript on objektorienteeritud keel, kus on dünaamilised andmetüübid ja asünkroonsed protsessid ning millel on palju teeke. Siiski võib väidetavalt koodi haldus raskeks minna suuremate projektide puhul [5], [6].

Python - väga populaarne ja objektiivselt kergelt õpitav. Sarnaselt javascriptile on ka python objektorienteeritud keel ning millele on tehtud palju teeke. Keele miinusteks on suhteliselt aeglane koodi käivitamine ja halb koodi haldus [5], [6].

Java - samuti väga populaarne ning on vanimate objektorienteeritud keelte seas. Java on tugevalt tüübitud, hea suuremate projektide jaoks, laseb mitut koodi ülesannet samal ajal käivitada ning on väga turvaline keel. Java aga vajab rohkem mälu kui teised programmeerimiskeeled [5], [6].

PHP - sarnaselt Pythonile on keel kergesti õpitav, seda saab kasutada nii teenuse- kui ka kliendipoolse osa jaoks, selle koodi ei kompileerita, vaid loetakse rea haaval. Keele üheks miinuseks on sarnaselt Javascriptile raske koodihaldus suuremate projektide puhul [5], [6].

Sarnaselt tehnoloogia valikutega saab ka siin iga keelega rakenduse tehtud. Antud lõputöö autor otsustas valida teenusepoolseks programmeerimiskeeleks Java, kuna sellel on head olemasolevad teegid ja raamistikud, kood on tugevalt tüübitud ning autoril on rohkem kogemusi selle keelega.

4.2.2 Teenusepoolsed raamistikud ja teegid

Valitud programmeerimiskeelele (Java) on tehtud palju teeke, mis pakuvad arendajale igasuguseid mugavusi koodi kirjutamise osas. Antud lõputöös kasutatakse peamiselt järgmisi raamistikke ja teeke:

Gradle - automaatse koodi ehitamise tööriist, mis teeb arendaja elu mugavamaks. Valida oli kahe suurema tööriista vahel - Gradle või Maven. Mõlemad on väga sarnased, täidavad samat ülesannet ja nendes ei esine suuri erinevusi [7]. Gradle valik tuli ainuüksi lõputöö autori kogemuse põhjal.

Springframework - väga populaarne Java raamistik, mis annab mugavust ja paindlikkust koodi kirjutamise osas näiteks koodi sõltuvuste süstimine, arhitektuuri kihistamine, turvalisus jms [8].

Lombok - Java teek mis aitab vähendada stereotüüpilist koodi annotatsioonide kaudu, näiteks genereerib automaatselt koodi konstruktorid, objektide võrdlemised jms [9].

JPA - eraldab koodi andmebaasist, millega arendaja ei pea kasutama tavalisi SQL lauseid, et andmebaasist infot kätte saada. Selle jaoks on JPA enda funktsioonid [10].

Jsoup - teek, mida kasutatakse veebi kraapimiseks. Sellega saab veebilehtedest kätte vajalikku infot ning seda muundada [11].

Peale nende on veel paar teeki, mis aitavad JSON objektide loomisega.

4.2.3 Kliendipoolse programmeerimiskeele valik

Domineeriv kliendipoolne programmeerimiskeel on Javascript ning sellel põhinevad keeled, seega pole peale selle midagi valida. Samuti on olulised ka HTML ja CSS mis pole iseenesest keeled vaid aitavad struktureerida ja kujundada veebilehti. Siiski võiks mainida, et plaanis on kasutada TypeScripti, mis on tugevalt tüübitud Javascripti peale üles ehitatud keel, seega pole see otseselt iseseisev programmeerimiskeel [12].

4.2.4 Kliendipoolsed raamistikud ja teegid

Kliendipoolseks raamistikuks või teegiks on saadaval palju valikuid. Populaarsemad neist on React, Angular, Vue, jQuery [12].

React - Facebooki poolt loodud Javascripti teek, mis on üks kõige populaarsemate kliendipoolse teekide ja raamistike seas. Kasutab JSX-i (JavaScript XML) ja virtuaalseid DOMi uuendusi, et mingi elemendi uuendamisel ei laeta kogu lehte uuesti.
Angular - Typescript-põhine raamistik, mis aitab kirjutada rohkem arusaadavamat koodi kasutades andmete sidumist. See on hea suuremate projektide jaoks, sest seda on lihtsam hallata [12].

Vue - raamistik, mis sarnaselt Reactile kasutab virtuaalseid DOMi uuendusi ning lisaks kasutab kahesuunalist andmete sidumist, mis muudab koodi väga paindlikuks [12].

jQuery - Javascripti teek, mis lihtsustab HTML-i muutmist, sündmuste käitlemist [12].

Antud lõputöö autor otsustas valida kliendipoolse osa tegemiseks Reacti koos Typescript'iga. Peale selle kasutatakse veel Axios teeki, mille ülesanne on lihtsustada päringute tegemist ja andmete saamist.

4.3 Andmebaasi valik

Tänapäeval on andmete hoidmise modernseks lahenduseks RDBMS andmebaasid. Sellised andmebaasid hoiavad andmeid tabelites ja kasutavad SQL lauseid andmete pärimiseks [13]. Hetkeseisuga neli kõige populaarsemat andmebaasi on Oracle, MySQL, Microsoft SQL Server ja PostgreSQL [14]. Järgnevalt kirjeldatakse mõned RDBMS andmebaasid ja tuuakse esile nende positiivseid ja negatiivseid küljed:

Oracle - üks suurimaid ja laialdasemalt kasutatavad ettevõtte tasemel andmebaase. Sellel on palju erinevaid andmetüüpe ja pakub kõrge taseme analüüsi, masinõpet ja palju muud. Siiski võib Oracle osutada kalliks ning võib olla raskesti üles seatav, seega pole see parim lahendus väiksemate rakenduste jaoks [14].

MySQL - vabavaraline andmebaas, mis toetab palju erinevaid andmetüüpe ja funktsioone. MySQL pole küll parim lahendus suuremate ja keerulisemate andmestruktuuridele, mis vajavad kõrge taseme analüüsimist või masinõpet, kuid see on lihtsasti kasutatav ning on kasvanud üheks kõige populaarsemaks andmebaasiks maailmas [14].

Microsoft SQL Server - sarnaselt Oracle andmebaasile on see väga populaarne ning laialdaselt kasutatud ettevõtetes. SQL Server pakub mitu erinevat väljaannet vastavalt vajadusele. Kuigi, nagu ka Oracle puhul, võib see suuremate projektide puhul minna kalliks maksma [15].

PostgreSQL - väga mitmekülgne ja paindlik andmebaas. See toetab mitmeid tänapäeval populaarseid programmeerimiskeeli [16].

Kõik on head valikud, töö saab tehtud iga andmebaasi valikuga ning antud lõputöö autor otsustas valida PostgreSQL.

4.4 Arenduskeskkonna valik

Programmeerimise lahutamatuks osaks on saanud hajutatud versioonikontrolli süsteem. Nimelt Git ületab kõiki teisi süsteeme ning seda kasutavad tuntud GitHub, GitLab ja BitBucket, mida me järgnevalt võrdleme:

GitHub - kõige levinud koodihaldus tööriist, millel on üle 100 miljoni kasutaja. Seda saab kasutada tasuta, mis on piiratud funktsionaalsusega, või osta enda arenduseks vastavalt parem pakett [17], [18].

GitLab - hetke seisuga üle 30 miljoni kasutaja, GitLab pakub natuke rohkem mugavusi ja teenuseid tasuta versioonis kui GitHub. Näiteks laseb GitLab tasuta versiooni kasutajal hoida repositoorium enda organisatsioonis [17], [19].

Bitbucket - populaarsust koguv tööriist, pakub odavamat teenust väiksema koodi mahule. Samuti on Bitbucket tehtud Atlassiani arendajate poolt, seega töötavad nende tehtud muud rakendused ja teenused (näiteks Jira) omavahel hästi [20].

Väiksema rakenduse jaoks on kõigil kolmel tasuta versioon saadaval, seega pole otseselt vahet, millist pakkujat võtta. Siiski peab arvestama, et kui on tulevikus edasi arenduseks vaja suuremat ja paremat teenust, siis oleks hea võtta kõige odavam neist.

Kuna bitbucket pakub kõige odavamat teenust võrreldes GitLab-i ja GitHub-ga otsustas antud lõputöö autor valida Bitbucket-i.

Programmeerimiseks on tänapäeval vaja ka head integreeritud arenduskeskkonda (IDE). Kuna antud lõputöös sai valitud kliendi- ja teenusepoolseteks programmeerimiskeelteks Java ja Javascript, peaks valitud arenduskeskkond neid ka toetama.
JetBrains-i loodud IntelliJ IDEA on hetkeseisuga kõige populaarsem Java IDE. Avatud lähtekoodiga ja kooli projektidele on see tasuta, seega saab seda ära kasutada [21].

Javascripti jaoks on aga üks populaarsematest IDE-dest Microsofti loodud Visual Studio Code. See toetab palju erinevaid programmeerimiskeeli ning saab vajadusel toetusi juurde installida. Pealegi on see tasuta [22].
Seega on antud lõputöös kasutatavateks arenduskeskkondadeks IntelliJ IDEA ja Visual Studio Code.

4.5 Arhitektuur

Veebirakenduse kogu struktuur koosneb andmebaasist, veebiteenusest ja klientrakendusest. Samuti on kaasatud versioonihaldus ja välised teegid.
Veebiteenuse ja klientrakenduse arendus eraldi, kuid mõlemad on omavahel seotud. Nii teenuse kui ka kliendipoolsel rakendusel on oma versioonihaldus ja välised teegid. Veebiteenuse puhul saab kasutada mälupõhist andmebaasi, kuhu saab andmed lisada olemasolevatele SQL skriptidele. Koodi muudatuste vajadusel saab versioonihaldusest alla tõmmata kõik vajalik kood ning teha muudatused uude harusse.

4.6 Veebirakenduse disain

Järgnevalt räägitakse lähemalt veebirakenduse disainist nii funktsionaalsete kui ja kujundlike aspektide poolest. Rakenduse disainimisel on fookus kasutajakogemusel ning soovitakse saavutada mängu stiiliga sarnane tulemus.

4.6.1 Kasutajakogemus

Kuna tegemist on lihtsama ühe lehe rakendusega, saab rohkem tähelepanu pöörata kasutajakogemusele. On fokuseeritud järgmistele funktsionaalsustele:

Kõik andmed laetakse korraga klientrakendusse, mis teeb rakenduse kiiremaks ning millega ei tehta üleliigselt päringuid teenusrakenduse pihta.

Otsinguriba ja filtreerimine töötab koheselt, peale ühe tähe kirjutamist või nupu vajutamist peaks otsitav nimekiri juba muutuma.

Rakenduse põhi funktsionaalsus on filtreerimine, seega ei lasta kliendil vajutada üleliigselt nuppe. Näiteks ei saa valida filtriteks mingit kehaosa/tüüpi koos efektiga, kui sellele kombinatsioonile ei ole olemas riideeset. Vastavalt filtreeritakse sellised nupud nimekirjast ära või tehakse halliks ehk mitte kasutatavaks.

4.6.2 Rakenduse disain

Disainiks kasutatakse mängule omapäraseid ja tuttavaid värve ning elemente. Mängus on nuppude ja taustade peamiseks värviks sinakas toon. Vajutavate nuppude disainiks on helesinised kastid ümarate nurkadega koos valge tekstiga.

Joonis 1. Growtopia mängus olevad nupud

Joonis 2. Mängus olev tekst ja värvid.

Mängus domineerivaks tagatausta värviks on sinakad toonid ning mängus kasutatav teksti font on Century Gothic Bold.

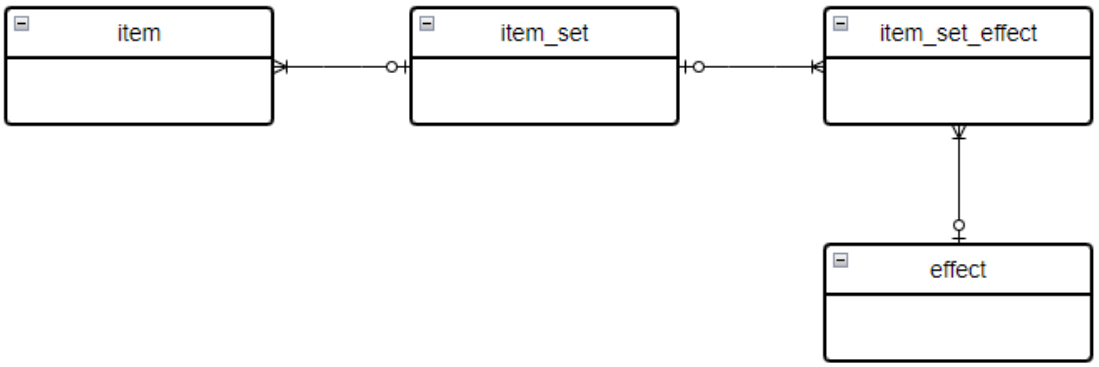
Antud lõputöö rakenduse põhitoonideks on sarnaselt mängule pandud sinakad värvid, täpsemalt tumedam sinakas toon tausta värviks ja helesinine koos valge tekstiga nuppude, filtrite jms värviks. Nende valikutega sarnaneb rakendus mänguga.

4.6.3 Andmebaasi disain

Kuna kõik mängud on erinevad, ei ole olemas üldist olemasolevat lahendust nagu on näiteks kasutajate juurdepääsuga, vaid tuleb vastavalt vajadusele luua tabelid ja seosed. Mängus on kehaosad, efektid ja riideesemed. Tehakse algselt nendele kolmele andmete hoidmiseks tabel. Kasutatakse inglise keelseid nimetusi, et tulevikus probleeme vältida. Kehaosa võib ka nimetada kui riideeseme tüüp. Seega on hetkel tabelid “type”, “item” ja “effect”. Kuna üks riideese võib anda mitu efekti, tuleks tabelite “effect” ja “item” teha vahetabel “item_effect”. Samuti esineb juhuseid, kus ühe efekti saamiseks läheb tarvis mitu riideeset. Seega tuleks tabelite loogikat vastavalt parandada. Tekib juurde tabel “item_set”, mis ühendab vastavad esemed üheks komplektiks. Tuleb ka arvestada sellega, et üks komplekt võib anda mitu efetki. Selle käigus muutub tabel “item_effect” “item_set_effect”-ks. Analüüsi käigus selgus, et tüüpe on kokku 10 ja need on mängu

10 aasta eksisteerimise käigus muutunud 1 korra, seega tüübi andmebaasist välja võtta ja hoida koodis enumina. See lihtsustab andmebaasist päringuid.

Paljudel efektidel on ka vastand efekt. Näiteks on olemas efektid, kus üks annab karakterile kiirust, teine aga teeb teda aeglasemaks. Efekte võib kujutada kui punkte. Kaks kiiruse efekti ja üks aegluse efekt annab kokku ühe kiiruse efekti, ehk 2 - 1 = 1. Kuidagi peab ka seda kajastama. Üheks lahenduseks on teha eraldi efekti tabelid näiteks "positiivne_efekt" ja "negatiivne_efekt" ning selliselt on vaja kuidagi vastand efektid ühendada. Teiseks lahenduseks on jätta üks efekti tabel ja panna mingi efekti vastandi olemasolul samasse tabelisse nii positiivne kui ka negatiivne efekt. Antud lõputöö autor otsustas valida viimase lahenduse. Selle käigus on *backend* pärimine kiirem ja saadetakse vähem infot, kuid *frontend* poole peal peab tegema vastavad muudatused, et nii positiivset kui ka negatiivset efekti eraldi näidata. Positiivset ja negatiivset efekti riideeseme komplektil näitab "item_set_effect" väli "amount" ehk kogus. Tavaliselt antakse mingit efekti 1 kord, kuid mõni komplekt võib anda mingit efekti rohkem kui teised, seega on igal efektil tehtud kogus, mis näitab kui palju mingit efekti komplekt annab. Peale selle tähistab negatiivne kogus negatiivset efekti ja positiivne positiivset efekti. Joonisel on kujutatud andmebaasi skeem lihtsustatud kujul. Täielik andmebaasi skeem on nähtav peatükis Lisa 2.



Joonis 3. Lihtsustatud olemi-suhte diagramm

4.7 Analüüsi kokkuvõte

Analüüsis sai paika pandud rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded, räägiti erinevatest veebirakenduse arenduse tehnoloogiatest, teenuse- ja kliendipoolse programmeerimiskeelte valikutest kui ka olemasolevatest ning kasutatavatest

raamistikest ning teekidest. Lisaks räägiti andmebaasi valikust, rakenduse jaoks vajaminevat andmebaasi skeemist ning rakenduse disainist.

Arenduse tehnoloogiaks valiti REST API lähenemine. Teenusepoolseks programeerimiskeeleks valiti Java ning raamistikeks ja teekideks Gradle, Springframework, Lombok, JPA, Jsoup. Kliendipoolseks programmeerimiskeeleks valiti Javascripti peal põhinev Typescript koos React teegiga. Andmebaasiks osutus PostgreSQL. Arenduskeskkondadeks valiti IntelliJ IDEA ja Visual Studio Code ning koodihalduseks Bitbucket.

5 Veebirakenduse arendus

Järgnevates peatükkides räägitakse veebirakenduse veebiteenuse- ja kliendi-poolsest arendusest vastavalt nõuetele.

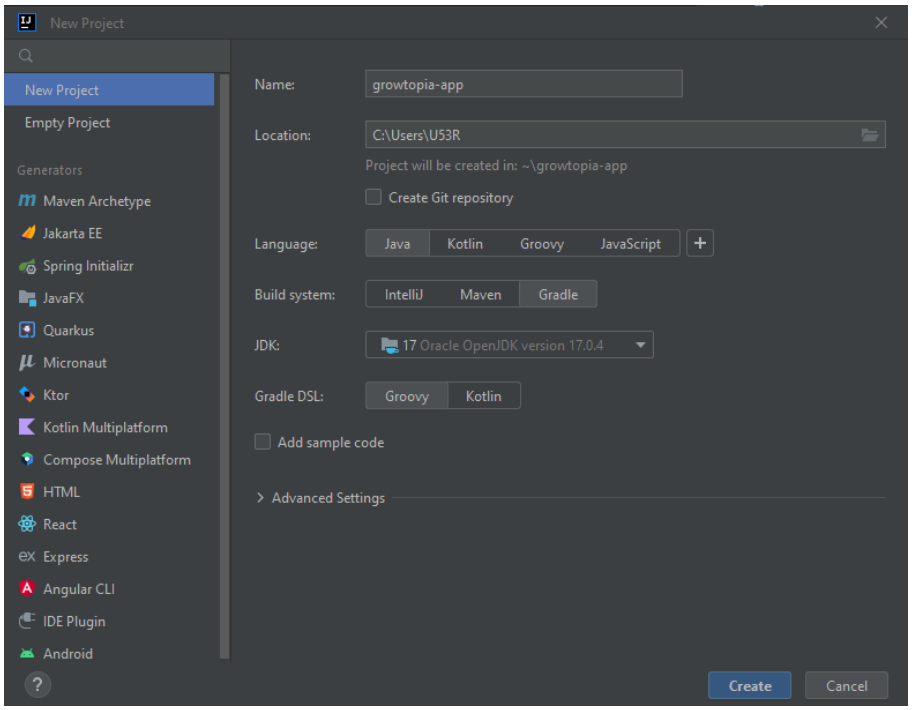
5.1 Veebiteenuse-poolne lahendus

Veebiteenuse-poolne rakendus tegeleb päringutega, andmebaasiga suhtlemisega ja andmete väljastamisega. Rakendus on ehitatud Spring Boot raamistikuga ning kasutab Postgre andmebaasi.

5.1.1 Spring Boot rakenduse arhitektuur

Spring boot on üles ehitatud spring raamistiku peal, tehes tava Spring raamistikuga võrreldes API loomise ja konfigureerimise lihtsamaks. Spring boot kasutab konfigureerimiseks annotatsioone [23].

Rakendus on loodud kasutades IntelliJ IDEA uue projekti loomise funktsionaalsust. Valides vastavad valikud “Java”, “Gradle” ja “Groovy” ehitab IDEA tooriku valmis.



Joonis 4. Projekti tooriku loomine IDEA kaudu

Rakenduse juurkausta tekkis src/main/java kuhu läheb enamus loodava rakenduse koodist. Tehakse juurde järgmised kaustad:

- src/main/java/config - kaustas on kood, mis tegeleb konfiguratsiooniga
- src/main/java/entities - kaustas on olemid, mida kasutatakse nii andmebaasi kirjete kui ka päringute jaoks JSON objektide loomisel.
- src/main/java/itemapi - kaustas on kood, mis tegeleb päringute ja andmete ligipääsuga
- src/main/java/scraper - kaustas on kood, mis tegeleb veebi kraapimisega andmete kättesaamiseks.
- src/main/resources kaustas on kõik sql. failid andmete sisestamiseks ning ka application.properties fail, kus hoitakse vajalikke andmebaasi väärtuseid.
- src/items kaustas on kõik teksti failid, mis luuakse peale veebikraapimist.

Juurkaustas on ka build.gradle fail - Gradle konfiguratsiooni fail. See hoolitseb projekti sõltuvuste ja väliste teekide eest. Lisaks on veel mõned Gradle jaoks vajalikud failid.

5.1.2 Andmebaasiga suhtlemine

Antud projektis kasutatakse andmebaasiga suhtlemiseks JPA-d ehk Java Persistence API koos Hibernate ORM-i raamistikuga. JPA annab andmebaasiga suhtlemiseks kõrgetasemelise abstraktsioonikihi, ning Hibernate pakub objektide kaardistamiseks võimsaid funktsioone. Neid tehnoloogiaid koos kasutades lihtsustatakse andmetele juurdepääsu ja vähendatakse andmebaasi suhtlemise jaoks vajalikku koodi hulka. Domeeniobjektidele pannakse peale erinevad annotatsioonid näiteks *@Entity* või *@Embeddable*. Nendega ühendatakse objektide omavahelised seosed ning tehakse objektid Hibernate'i jaoks tõlgentatavaks.

JPA peab ka teadma, millise andmebaasiga on tegemist. Seda saab JPA konfiguratsioonis täpsustada. Teenuse seadistused kirjeldatakse konfiguratsiooni failides ja klassides, mida Spring rakendus tunneb ära tänu annotatsioonidele - *@Configuration*. Seal määratakse andmebaasi tüüp, sellega ühendamiseks vajalikud väärtused ning kõik failid, kust otsida objekte.

Arenduse käigus saab vajadusel kasutada ka mälupõhist hsqldb andmebaasi.

5.1.3 REST API päringud

REST API päringute loomisel on kasutatud nimesid, mis kirjeldavad päritavaid andmeid. Ehk “types” päring tagastab tüübid ehk kehaosad, “effects” efektid ja “items” riideesemed. Samuti on tihtipeale tavaks panna ressursside ette */api*.
Võimalikud tagastuskoodid on 200, 400, 404, 500.

Tabel 1. Rakenduse API päringud

Päringu kirjeldus	Päringu tüüp	Ressurss
Kehaosade ehk tüüpide pärimine	GET	/api/types
Efektide pärimine	GET	/api/effects
Riideesemete pärimine	GET	/api/items

5.2 Lähteandmed ja veebikraapimine

Rakendus töötamiseks läheb vaja vastavaid lähteandmeid. Nende kättesaamiseks kasutatakse vikipeedia veebikraapimist Jsoup teegi abil.

5.2.1 Kehaosade ehk tüüpide ja efektide andmed

Vaja läheb kõiki kehaosi ehk tüüpe, riideesemeid ja nendele olemasolevaid efekte, mis on mängu vikipeedias olemas. Tüüpe on hetke seisuga mängus kaheksa. Need saab lisada käsitsi koodi enumiks.
Efekte leiab erinevatest vikipeedia nimekirjadest kui otsida näiteks sõna *mod*, kuid nendes on välja toodud ainult enamlevinud efektid. Kuna vikipeedias pole kõiki olemasolevaid efekte ühes kohas nimekirjana kirjas ja mõnedel efektidel on mitu nimetust, peab need käsitsi leidma ja nendest andmebaasi SQL laused tegema. Teades enam-vähem mängu ja nende efekte, saab enamlevinud efektid kirja panna, ning vähem tuntud efektid leida veebikraapimise käigus. Mõnedel efektidel on vastandid, näiteks kiiremini liikumine ja aeglasemalt liikumine, mis pannakse üheks efekti objektiks

kokku. Enamus vastandid on iseenesest mõistetavad, näiteks “löögi tugevus” ja “löögi nõrkus”, “eemale lükkamine” ja “tõmbamine” jne.

5.2.2 Riideesemete info kraapimine

Nõuete kohaselt on vaja leida riideesemed ja nende info. Täpsemalt nimetus, tüüp, kas ese on vahetatav teiste mängijatega, kas eseme välimust saab muuta ning mis efektid sellel on. Kogu info on iga riideeseme jaoks pühendatud vikipeedia lehel, seega tuleb kõik need lehed veebikraapides läbi käia.
Kõiki riideesemeid saab leida vikipeedia igasugustes nimekirjades. Näiteks on olemas nimekiri, kus on loetletud absoluutselt kõik riideesemed. Samuti on ka kehaosa ehk tüübi järgi grupeeritud nimekirjad, mida saab antud projektis ära kasutada. Nende kaudu on lihtsam teha riideesemete jaoks andmebaasi kirjed - kui on tüübi järgi grupeeritud, siis ei pea hakkama riideeseme tüüpi eraldi lehelt otsima. Nii saab vajadusel ka tükeldada veebikraapimise programmi.
Veebi kraapija võtab ette iga tüübi kohta tehtud riideesemete nimekirja URL-i, otsib iga riideeseme URL-i ja töötab selle läbi. Kraapiga otsib nimekirjas ka “next” nuppu ehk linki järgmisele leheküljele, kui kõik esemed ei mahtunud ühele lehele ära.

Enamus riideesemete info lehtedel on sarnane struktuur, kuid nendes erineb erandeid, seega on raske kirjutada programm, mis saab iga eseme koos efektiga vigadeta kätte.
Riideeseme info lehe vikipeedias võib jagada kolmeks: riideeseme nimetus, omadused ja kirjeldus. Nimetus ja omadused on alati kindlas kohas ja sama struktuuriga, kuid kirjelduse oma võib erineda. Eseme nime, tüübi, vahetatavuse, välimuse muutmise võimalus ja mõne üksiku efekti saab nimetuse ja muu info alt kätte, enamik efektid on kirjutatud kirjelduse alla vabas vormis. Seega peab enamus efektide kättesaamiseks otsima inimeste poolt kirjutatud lausetest märksõnu, mis viitab efekti olemasolule riideesemel.
Veebikraapija otsib teksti HTML elemendi ja/või klassi järgi. Riideeseme nimetuse leiab <*b*> HTML elemendi seest, mis on <*span*> HTML elemendi sees, millel on klass *mw-headline*. Omadused leiab <*div*> HTML elemendi seest mille klass on *card-text*. Neid elemente sama klassi nimega on küll rohkem kui üks aga tavaliselt on neid ainult kaks ning vajalik info asub teises elemendis. Seega saab korjata kokku kõik need elemendid selle klassiga ning otsida infot ainult teisest elemendist.

Kõige tülikam on efektide otsimine riideeseme kirjeldusest. Samuti saab sealt teada, kas mingi ese on efektiga või mitte, mis on vajalik kuna nõuete kohaselt ei kasutata esemeid, millel pole efekte.
Riideeseme kirjeldus võib lehel olla erinevates kohtades. Täpsemalt, on kirjeldus tavaliselt <*p*> HTML elemendi ehk lõigu sees, kuid lõikude arv on tihtipeale erinev.

Veebikraapimiseks on efektidele tehtud positiivne nimi ja negatiivne nimi ning mõlemale on antud omad sünonüümid, mille järgi neid lausetest otsida. Näiteks on kiiruse efektil positiivne nimi "Speedy" kuid selle väljendi puudumisel võib lausetest otsida "run faster", "move faster". Selle negatiivseks efektiks on "Slow" ja selle sünonüümid "run slow", "move slow", "walk slow". Kui kraapija leiab vastava sõna kirjeldusest, tehakse koodis riideeseme ja efekti vahel seos ning vastavalt pannakse vaikimisi kogus ("amount") positiivse puhul 1 ja negatiivse puhul -1.
Kui kogu riideeseme info on läbi käidud ja leitud vastavad efektid, tehakse esemest SQL laused.

5.2.3 Kraabitud info töötlemine

Kui kõik läheb hästi tehakse peale veebikraapimist kogutud andmetest SQL laused. Need tuleks siiski inimese poolt üle vaadata, mitte kohe andmebaasi panna. Selleks pannakse kõik SQL laused ühte faili kokku.
Veebikraapimise käigus võib veel juhtuda palju erinevaid olukordi. Näiteks ei leita üles eseme nime, omadusi või kirjeldust, ei leita efekte, URL võib olla midagi muud kui riideeseme infoleht jms. Nende jaoks on tehtud erinevad kontrollid ning probleemsed või kontrollimist vajavad URL-id pannakse vastavatesse failidesse ülevaatamiseks.
Efektidega riideesemeid eristab tavalistest esemetest sõna "mod" kirjelduses. Sellega saab kindlaks teha, et mingil esemel on efekt olemas. Kui ei leita vastavat efekti, pannakse eseme URL koos viga mainivat sõna tekstifaili kirja.
Järgmisena on välja toodud erinevad failid ning missugused URL-id ja info nendesse läheb:

- items.sql - siia faili pannakse kõik efektidega esemed, mis lähevad lõpuks andmebaasi. Seal tehakse vajadusel parandused riideesemete SQL lausetele. Riideesemest tehakse SQL laused juhul kui leitakse selle kirjeldusest sõna

“mod” ning andmebaasist leitakse vähemalt üks efekt, mida on kirjelduses mainitud.

- cosmetic.txt - siia faili pannakse kõik esemed, millel on “cosmetic” või “aesthetic” sõna sees. Kuna projektis ei kasutata funktsionaalsete efektideta riideesemeid, siis nendel on tavaliselt kirjelduses lause “serves a fully cosmetic/aesthetic purpose”. Siiski pannakse need faili igaks juhuks ülevaatamiseks.
- error.txt - siia faili pannakse kõik esemed, mille kraapimisel esines viga. Faili pannakse eseme URL, info (juhul kui on olemas näiteks nimetus ja kirjeldus) ja veateade. Riideese on vigane kui selle nime ei leitud, selle kirjeldust ei leitud või ei leitud andmebaasis olevatest efektidest vastet, kuigi kirjelduses oli sõna “mod” sees. Nii saab leida ka uusi efekte, mida varem andmebaasis polnud.
- check.txt - siia faili pannakse kõik esemed, mis on pandud items.sql faili, kuid mida tuleks üle vaadata. Siia lähevad kirjed, kui riideeseme kirjeldus sisaldab teatuid sõnu.

Sõnad mille alusel pannakse ese check.txt faili on järgmised:

- "with" - sõna on tõlgituna “koos”, mis võib viidata sellele, et riideese vajab mingit teist riiet, et efekti saada. Hetkel pole head viisi neid koodis kokku sobitada ning selliseid esemeid on vähe. Seega saab käsitsi parandades need korda.
- "slippery" - mõned esemed annavad efekti mängus libiseda igal pinnal ning mõni ei lase libiseda. kuna mõlemad efektid kasutavad sama sõna, peab üle vaata, mis efekt mingil esemel on.
- "slide" - v.t “slippery” (efekti sünonüüm)
- "comet" - efekti nimetus mis on mitme riideeseme nimes ja kirjelduses, kuid ei anna antud efekti.
- "instead" - mõnel riideeseme kirjelduses on sõna “hoopis” sees, mis võib mainida mingi efekti mitte olemasolu. Nende esemete SQL lausetes võib olla üleliigseid või valesid efekte. Kuna kirjeldused on kirjutatud vabas vormis võib näiteks olla öeldud “ese ei anna … efekti vaid hoopis … efekti.

5.3 Klientrakenduse-poolne lahendus

Kui veebiteenuse-poolne rakendus tegeleb ainult andmebaasiga suhtlemisega ja andmete väljastamisega, toimub suurem osa tööst kliendipoolses rakenduses. Siin saab otsida, filtreerida ja riietada visuaalset karakteri.

5.3.1 React rakenduse struktuur

Kasutades NPM-i saame luua React rakenduse kirjutades konsooli järgmise käsu:
“npx create-react-app growtopia-app --template typescript --use-npm”
Sellega tehakse valmis React rakendus nimega “growtopia-app”, mida saab kohe peale projekti loomist tööle panna käsuga “npm start”.
Projekti loomisel tekivad igasugused kaustad ja failid, et see töötaks. Tähtsamad neist on:

- node_modules kaust - siin on kõik Reacti ja installitud sõltuvuste kood.
- public ja src kaustad - siin on algselt genereeritud projekti lehe kood ning kuhu tuleb enamus arendatavast koodist.
- package.json ja package-lock.json failid - projekti konfiguratsiooni failid, kus on kirjas sõltuvused, versioonid jms.
- index.tsx - React Typescripti fail - juurfail
- MainApp.tsx fail - React Typescripti fail, kuhu saab koodi kirjutada. Antud projekti ühe lehe rakenduse funktsionaalsused on selles failis.

Tehakse juurde järgmised kaustad:

- src/components - siin kaustas on erinevad komponendid, mida saab taaskasutada, näiteks veebilehe päis ja jalus.
- src/domain - siin kaustas on kõik klassid, mis on abiks päringul saadud JSON andmete muundamine Typescripti objektideks.
- src/services - siin kaustas on kõik klassid, mis tegelevad päringutega.

5.3.2 Suhtlus veebiteenusega

Veebiteenusega suhtlemiseks kasutatakse Axios teeki. Tüübid, efektid ja riideesemed saadakse veebiteenuse poolsest rakenduse vastavast API-st pärides. Axios kasutab *promise* objekti asünkroonsete toimingute haldamiseks, antud juhul päringute tegemisel. Kuna peale päringu tegemist läheb veidi aega enne kui andmed kohale jõuavad ootab

promise päringu vastuse ära enne vastava koodi käivitamist. Kõik andmed päritakse üks kord lehe laadimisel ning muundatakse “service” kaustas olevateks vastavateks typescripti objektideks ning hoitakse mälus. Nii et tehta üleliigseid päringuid *backend*-i pihta.

5.3.3 Rakenduse kasutamine

Rakendus kasutab ühe-lehe-rakenduse põhimõtet, kus kõik funktsionaalsused ja info on ühe lehe peal. Lehel kuvatakse korraga kõik kehaosad ehk tüübid, efektid ja riideesemed. Pärimise käigus saadakse riideesemed komplektina millesse kuuluvad riideesemed, kuid rakenduses kuvatakse riideesemed eraldi, mitte komplektina. Kuna efektide arv on 50-100 vahemikus ja riideesemed on üle 1000, on tehtud neile otsimise võimalus. Nimekirjadel on otsinguriba, mis otsib kohe peale tähe kirjutamist efekte või riideesemeid nime järgi. Tüüpe on hetkeseisuga kokku 10 ning nendele pole vaja otsimise võimalust kuna kõik need mahuvad lehele ära. Samuti on lehel ka simuleeritud karakter, kellele saab iga kehaosale panna mingi riideeseme, millel on sama tüüp. Simuleeritud karakter näitab, mis efekte mängus olev karakter saaks kui ta kannaks valitud riideesemeid. Saadavad efektid kuvatakse ka automaatselt niipea kui pannakse peale, vahetatakse või eemaldatakse riideese simuleeritud karakterilt.

Kasutaja saab üle 1000 riideeseme filtreerida peale otsinguriba veel tüübi ja efektide järgi. Vajutades mingi tüübi peale tüübi loetelus näidatakse riideesemete nimekirjas ainult neid esemeid, millel on valitud tüüp. Kuna riideesemel saab olla ainult üks tüüp, ei lasta kasutajal valida mitu tüüpi korraga, sest nii ei näidata ühtegi riideesemet nimekirjas. Efektide filtrid töötavad sarnaselt peale selle, et efekte saab valida rohkem kui üks, kuna mingi ese võib anda mitu efekti. Peale efekti ja/või tüübi filtriks valimist filtreeritakse ka efektid vastavalt riideesemete olemasolule. Rakendus käitub nii, et kasutaja ei saa valida korraga neid filtreid, mis teeks riideesemete loetelu tühjaks. Mingite tüübi ja efektide filtrite valimisel on alati riideesemete nimekirjas vähemalt üks ese.

Efektide ja riideesemete info kuvatakse *tooltip*-na, mida saab näha hiire hõljumisel nende peal. Nõuetekohaselt näidatakse efektide hõljumisel kirjeldust ja riideesemetel näidatakse selle kandmisel saadavaid efekte, kas ese on vahetatav, muudetav või vajab teisi esemeid efekti saamiseks. Viimased kolm on märgitud ka riideesemete nimekirjas olevatel kirjetel. Nii saab kasutaja näha osa infot hiirega eseme peal hõljumata.

5.3.4 Rakenduse tehniline pool

Kuna antud veebirakenduse peamine osa on otsimine ja filtreerimine erinevate parameetrite põhjal ning riideesemete nimekiri on üle 1000 objekti pikk, tuleks pöörata rohkem tähelepanu jõudlusele. Selle saavutamist on kirjeldatud järgnevalt.

Andmete pärimisel saadakse esemete komplektid, mis sisaldavad nii esemeid kui ka efekte. Esemete kuvamise lihtsustamiseks on tehtud lisaks komplektide hoidmisele ka iga eseme jaoks eraldi nimekiri. Nii komplektide kui ka esemete nimekirjad on vajalikud erinevate filtreerimis ja otsimise funktsioonideks. Samuti on tehtud eraldi objektideks ühe efekti positiivne ja negatiivne osa. Nendega küll suurendatakse rakenduse jaoks vajaminevat mälumahtu, kuid vähendatakse jõudluse mahtu.

Reacti üks positiivseid omadusi on see, et igat komponenti saab eraldi formuleerida, ehk kui lehel üks komponent muutub pole vaja lehe laadimisel igat teist komponenti uuesti laadida. Selle saavutamiseks kasutatakse React *hooks*-e. Seda saab antud projektis ära kasutada pannes iga nimekirja *hook*-i kasutama . Nii ei pea näiteks efektide nimekirja uuesti laadima, kui kasutatakse riideesemete otsimiseks selle otsinguriba. Nimekirjad laetakse uuesti ainult siis kui nendes midagi muutub.

5.3.5 Rakenduse kujundlik pool

React rakenduses saab stiilifailid (*.css* lõpuga) faile importida *.tsx* failidesse. Stiilid pannakse failis HTML elementide või nendele pandud klassinimede järgi. Nii saab kujundada ning muuta lehe välimust vastavalt nõuetele.

Kuna efektide arv on 50-100 vahemikus ja riideesemed on üle 1000, ei mahu kõik need korraga lehele ära. Seega on rakenduses on nende nimekirjad fikseeritud suuruses ja igal nimekirjal on tehtud eraldi kerimise võimalus. Kõik nimekirjad ja simuleeritud karakter on pandud üksteise kõrvale, et mobiilis oleks lihtsam eristada nimekirjade vahel kui ka nende sees kerimist.

Rakenduses valitud filtritel s.t peale vajutatud tüüpidel ja efektidel on ümber ääris, mis teeb kasutajale nähtavaks, missugused filtrid on parasjagu valitud. Kasutajale ei kuvata efekte, mida ei saa filtriks valida, ehk mille valimisel poleks filtrite kombinatsioonil ühtegi riideeset. Tüübid ehk kehaosad aga, millele pole valitud efektiga ühtegi riideeset, tehakse nii öelda halliks ehk kasutamatuks.

Riideesemetel näidatakse nimekirjas, kas ese on vahetatav, muudetav või on osa komplektis eri värvide ja sümbolitega ringikujulised indikaatorid. Sümboliteks on

võetud vastava omaduse inglise keelne esimene täht. Indikaatoriteks on vastavalt sümbolid “U”, “T” ja “S”, mille inglisekeelsed väljendid on vastavalt “untradeable”, “transmutable” ning “set”. Värvideks on valitud vastavalt punane, roheline ja kollane. Värvid kajastuvad ja eseme *tooltip*-il.

5.4 Testimine

Antud lõputöö on suhteliselt lihtne ühe-lehe-rakendus, seega oleks automaattestide tegemine kulukam kui manuaalselt testimine. Samuti on rakenduse põhi eesmärgiks otsimine, filtreerimine ja info kuvamine, mida oleks automaattestidega tülikam ja rohkem aeganõudev teha. Pealegi saab manuaaltestimisega üle vaadata kasutajakogemuse, mis on antud rakenduse puhul tähtis [24]. Antud rakenduse testimise üheks positiivseks eeliseks on see, et *backend* ja *frontend* on eraldi seega saab neid eraldi testida.

5.4.1 Teenusepoolne manuaalne testimine

Teenusepoolses rakenduses tuleb testida andmete pärimist. Kõige lihtsam on panna *backend* püsti ja teha päringuid selle pihta. Testida tuleks kõik ressurssid (v.t peatükk 4.1.3). Antud lõputöös kasutatakse päringute tegemiseks Postman rakendust. Sellega vaadata kas kõik andmed on õiged.

5.4.2 Kliendipoolne manuaalne testimine

Suurem testimine toimub kliendipoolses rakenduses. Siin on vaja testida filtreid, otsimist, info kuvamist ja visuaalse karakteri toimimist
Seega on kirja pandud vastavad testid, mida tuleks manuaalselt läbi proovida:
Rakenduses on vaja testida või kontrollida järgmisi olukordi:

- lehel kuvatakse kõiki tüüpe, kõiki efekte, kõiki esemeid
- efektide ja riideesemete otsinguriba töötab korrektselt
- valitud kehaosal ja efektil on ümber mitte valitud omadest eristatav ääris
- valitud kehaosale ja/või efektile näidatakse õigeid efekte ja riideesemeid
- kui valitud kehaosal pole riideesemetega seoses mingisugust efekti siis seda ei kuvata filtri valikuna
- kui valitud efektil pole riideesemetega seoses mingisugust kehaosa siis see kehaosa on kasutamatu ehk tehtud halliks

- filtrid ja otsinguriba peavad töötama koos korrektselt
- simuleeritud karakterile saab riideesemeid lisada, vahetada ja eemaldada
- simuleeritud karakteril on näha õigeid efekte, kontrollida efekti saamist üksiku riideeseme ja kompletki korral.
- efektide ja riideesemete info kuvatakse õigesti
- riideesemetel on nimekirjas vahetatavuse, välimuse muutmise ja kompletki indikaatorid.
- riideesemete infolehel on korrektsed efektid ja indikaatoritele vastav info
- leht ei lähe katki mobiilivaates

6 Hinnang loodud veebirakendusele

Valminud veebirakendust võib hinnata saavutatud tulemuse ja nõuetele vastavuse alusel. Lisaks nendele võib hinnang kujuneda ka visuaalse disaini ja rakenduse edasise arendatavuse osas.

6.1 Saavutatud kasutatavus

Rakendus vastab nõuetele ja teeb seda milleks see loodud oli. Filtrid ja otsimine töötab korrektselt, on näha efektide ja riideesemete info ning saab nii öelda riietada simuleeritud karakteri. Samuti sai algandmed suures osas veebikraapimisega kätte. Siiski võib esineda vigu mõnede riideesemete ja sellele määratud efektide seoste vahel. Samuti on veel arenguruumi lehe stiili osas, eriti mobiilivaates ning nii teenuse kui ka kliendipoolset osa annaks optimeerida.

6.2 Võimalused edasi arenduseks

Rakendus kattis ainult antud lõputöös määratud nõuded, seega edasi arendamise võimalusi on palju. Järgnevalt on välja toodud mõtteid, kuidas arendust jätkata

1. Esiteks võiks valminud rakenduse liidestada teiste olemasolevate mängule tehtud rakendustega, mis laseks mängijatel hõlpsasti otsida, filtreerida efekte ja esemeid, näha nende infot, hinda ja visuaalset poolt karakteril. Nii peaks mängija kasutama kolme erineva rakenduse asemel ühte.
2. Teiseks on mängus olemas ka mõningad plokid ja ühekordse kasutusega esemed, millel on omad efektid. Rakendusse saaks teha nendele oma sektsiooni sarnaselt riideesemete omaga. Ära saaks kasutada olemasolevaid efekte ning vajadusel uued juurde teha.
3. Kolmandaks saaks juurde teha admin lehe, mis teeks üksikute efektide ja riideesemete lisamise ja muutmise lihtsamaks. Hetkeseisuga toimub see otse andmebaasis.
4. Neljandaks võiks andmete kraapimiseks kasutada tehisintellekti. Seda saaks välja õpetada vikipeedia lehtedelt infot lugema ning tegema vastavaid seoseid riideesemete ja efektide vahel.

7 Kokkuvõte

Antud lõputöö eesmärgiks oli luua Growtopia mängule riideesemete efektide planeerimiseks veebirakendus, mis aitaks mängijatel avastada ja planeerida uusi efekte ja riideesemeid.

Rakendus kasutab mängule sarnaseid elemente ja värve ning selles saab otsida, filtreerida efekte ja riideesemeid ning lugeda nende kohta infot. Samuti saab kasutaja riietada simuleeritud karakteri ja näha, mis efekte ta neid kandes saaks.

Töö analüüsis sai kirjeldatud erinevatest võimalikest tehnoloogiatest, programmeerimiskeeltest, andmebaasi valikutest ja rakenduse disainist. Töö arenduse peatükkides sail lähemalt räägitud koodi struktuurist, lähteandmete saamistest veebikraapimise teel, rakenduse kasutusest ning tehnilistest poolest kui ka disainist.

Antud bakalaureusetöö võib lugeda õnnestunuks, kuna rakendus töötab vastavalt nõuetele. Siiski saab rakendust veel paljuski edasi arendada nagu on välja toodud edasi arendamise peatükis. Teenuse- ja kliendi poolse osa eraldamine aitab suuresti kaasa tuleviku arendustele.

Kasutatud kirjandus

[1] Carmichael, C. (2019, mai 24). How to Make Money From a Website | 8 Simple Solutions. Website Builder Expert. https://www.websitebuilderexpert.com/building-websites/how-to-make-money-from-a-website/ [Külastatud 28 märts 2023]

[2] 2022 State of the API Report | API Technologies. (n.d.). Postman API Platform. https://www.postman.com/state-of-api/api-technologies/ [Külastatud 28 märts 2023]

[3] Simpson, J. (2023, veebruar 9). Top Architectural Styles for APIs in 2023 | Nordic APIs |. Nordic APIs. https://nordicapis.com/top-architectural-styles-for-apis-in-2023/ [Külastatud 28 märts 2023]

[4] Introduction to GraphQL. (2020, mai 19). CircleCI. https://circleci.com/blog/introduction-to-graphql/ [Külastatud 28 märts 2023]

[5] The Best 10 Backend Programming Languages. (2021, juuni 12). Back4App Blog. https://blog.back4app.com/backend-programming-languages-list/ [Külastatud 28 märts 2023]

[6] The Best Ten Backend Languages | Which is the best? (2021, veebruar 27). Back4App Blog. https://blog.back4app.com/best-backend-language/ [Külastatud 28 märts 2023]

[7] Altvater, A. (2017, juuni 29). Gradle vs. Maven: Performance, Compatibility, Speed, & Builds. Stackify. https://stackify.com/gradle-vs-maven/ [Külastatud 29 märts 2023]

[8] Spring Framework - Overview. (n.d.). Spring Framework - Overview. https://www.tutorialspoint.com/spring/spring_overview.htm [Külastatud 29 märts 2023]

[9] Lombok Java - Javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/lombok-java [Külastatud 29 märts 2023]

[10] Tyson, M. (n.d.). What is JPA? Introduction to Java persistence. InfoWorld. https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html [Külastatud 29 märts 2023]

[11] Hedley, J. (n.d.). jsoup: Java HTML parser, built for HTML editing, cleaning, scraping, and XSS safety. Jsoup: Java HTML Parser, Built for HTML Editing, Cleaning, Scraping, and XSS Safety. https://jsoup.org/ [Külastatud 29 märts 2023]

[12] 8 Best Front End Programming Languages | Simplilearn. (n.d.). Simplilearn.com. https://www.simplilearn.com/tutorials/programming-tutorial/best-front-end-programming-languages [Külastatud 29 märts 2023]

[13] What is a RDBMS (Relational Database Management System)? (2019, november

1). Data Management.
https://www.techtarget.com/searchdatamanagement/definition/RDBMS-relational-database-management-system [Külastatud 30 märts 2023]

[14] Chand, M. (2022, detsember 18). Most Popular Databases In The World (2023). Most Popular Databases in the World (2023).
https://www.c-sharpcorner.com/article/what-is-the-most-popular-database-in-the-world/ [Külastatud 30 märts 2023]

[15] L. (2019, jaanuar 17). Microsoft SQL Server Pros and Cons. LearnSQL.com.
https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/
[Külastatud 30 märts 2023]

[16] PostgreSQL: a closer look at the object-relational database management system. (2018, september 28). IONOS Digital Guide.
https://www.ionos.com/digitalguide/server/know-how/postgresql/
[Külastatud 30 märts 2023]

[17] GitHub vs GitLab: Difference Between GitHub and GitLab | upGrad blog. (2022, september 8). upGrad Blog.
https://www.upgrad.com/blog/github-vs-gitlab-difference-between-github-and-gitlab/ [Külastatud 31 märts 2023]

[18] Dohmke, T. (2023, jaanuar 25). 100 million developers and counting | The GitHub Blog. The GitHub Blog.
https://github.blog/2023-01-25-100-million-developers-and-counting/
[Külastatud 31 märts 2023]

[19] About GitLab. (2011, jaanuar 1). About GitLab | GitLab.
https://about.gitlab.com/company/ [Külastatud 31 märts 2023]

[20] A. (n.d.). Bitbucket - Pricing | Atlassian. Atlassian.
https://bitbucket.org/product/pricing [Külastatud 31 märts 2023]

[21] 10 Best Java IDEs in 2023 | Ultimate Guide to the Best Java IDE. (n.d.). Hackr.io.
https://hackr.io/blog/best-java-ides [Külastatud 13 aprill 2023]

[22] Which is the best IDE for JavaScript development in 2023? (2021, mai 17). DEV Community.
https://dev.to/theme_selection/what-is-the-best-ide-for-javascript-development-in-2021-1pmn [Külastatud 13 aprill 2023]

[23] What is Java Spring Boot? | IBM. (n.d.). What Is Java Spring Boot? | IBM.
https://www.ibm.com/topics/java-spring-boot [Külastatud 13 aprill 2023]

[24] J., & T. (2022, detsember 20). What Test Cases Cannot Be Automated and Why | Test Guild. Automation Testing Made Easy Tools Tips & Training.
https://testguild.com/what-to-automate/ [Külastatud 20 aprill 2023]

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

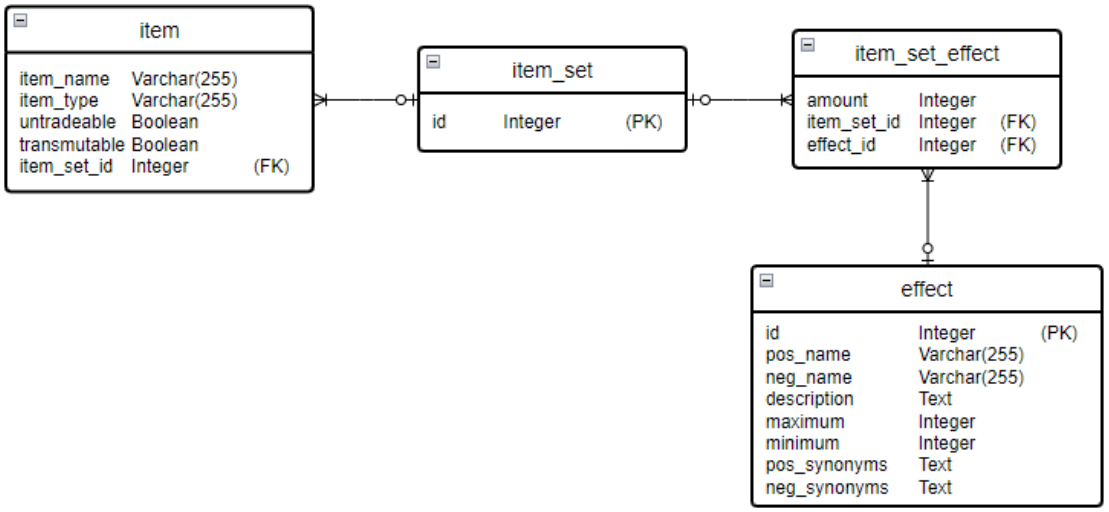
Mina, Reimo Jaansalu

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Karakteri riietusesemete efektide planeerija veebimängule", mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

05.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Olemi-suhte diagramm



Lisa 3 – Rakenduse kasutamise vaade